

Textbasierte Programmierung: Übungen zu Python

Allgemeine Information

Die folgenden Aufgaben beschäftigen sich mit **Verzweigungen** (if-Anweisungen) und **Wiederholungen** (for-Schleifen) in Python. Die Konzepte werden zunächst einzeln verwendet und in späteren Aufgaben kombiniert.

Zu jeder Aufgabe findest du Lösungsvorschläge in Form von Quellcode auf **OSS** im entsprechenden Ordner. Versuche die Aufgaben zunächst eigenständig zu bearbeiten und vergleiche anschließend mit dem Lösungsvorschlag. Oftmals gibt es mehr als eine korrekte Lösung.

Verzweigungen

A1

- a) Betrachte das folgende Programm und erkläre kurz seine Funktionsweise.

```
Sonne = input ('Scheint draußen die Sonne? (j/n)')
if Sonne == 'j':
    print('Denk an die Sonnenbrille!')
```

- b) Erweitere das Programm aus Aufgabenteil a), sodass auch für die Antwort 'n' eine Ausgabe generiert wird.

***Tipp:** Ergänze nach dem if-Fall einen else-Fall.*

- c) In vielen Anwendungen ist es sinnvoll, mehr als zwei Fälle abfragen zu können. Hierzu werden eine oder mehrere elif-Abfragen zwischen if- und else-Abfrage eingebaut. In jeder elif-Abfrage kann ein weiterer Fall abgefragt werden.

Übernimm den folgenden Code und ergänze sinnvolle Ausgaben. Ergänze anschließend Abfragen für weitere Wettererscheinungen.

```
Wetter = input ('Wie ist das Wetter heute? (Regen/Schnee/Sonne)')
if Wetter == 'Regen':
    #TODO Ausgabe
elif Wetter == 'Schnee':
    #TODO Ausgabe
else:
    #TODO Ausgabe
```

- d) Schreibe selbst ein Programm, dass mindestens drei verschiedene Eingabe unterscheiden kann und entsprechende Ausgaben generiert. Orientiere dich an Aufgabenteil c).

A2

- a) Betrachte folgenden Code. Welche Ausgabe wird von der Turtle generiert?

```
from gturtle import*
makeTurtle()

repeat:
    setRandomPos(300,300)
    if getX() < 0:
        setPenColor('RED')
        dot(10)
    else:
        setPenColor('GREEN')
        dot(10)
```

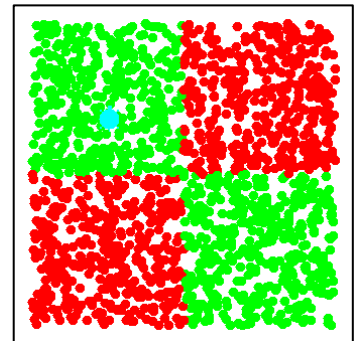
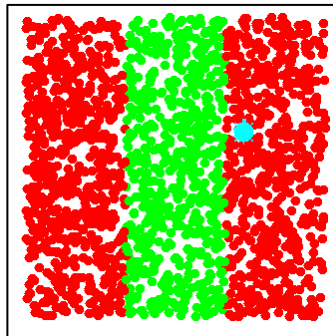
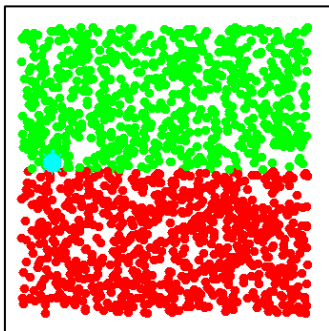
setRandomPos(300, 300) setzt die Turtle auf eine zufällige Position im Turtlefenster. Die x- und y-Koordinate liegen jeweils in einem Rechteck mit den Seitenlängen 300. Sie nehmen Werte zwischen -150 und 150 an.

getX() liefert die aktuelle x-Koordinate der Turtle. In diesem Fall ist es eine Zahl zwischen -150 und 150.

getY() liefert die aktuelle y-Koordinate der Turtle. In diesem Fall ist es eine Zahl zwischen -150 und 150.

Hinweis: Schlage dir unbekannte Funktionen nach. Zum Beispiel in der [Dokumentation](https://www.python-online.ch/turtlegrafik/) auf dieser Website: <https://www.python-online.ch/turtlegrafik/>

- b) Passe den Code aus Aufgabenteil a) jeweils so an, dass folgende Grafiken entstehen. Speichere jedes Programm einzeln ab.

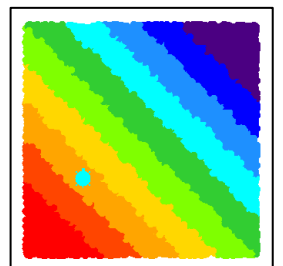


Hinweis: Denk dir in den Grafiken jeweils ein Koordinatensystem, dessen Ursprung genau in der Mitte liegt. Entscheide dann für welche x- bzw. y-Koordinaten die Farbe Grün gewählt wird. Kombiniere Bedingungen mit and oder or.

- c) Schreibe ein Programm, das einen Regenbogen / Farbverlauf erzeugt. Horizontale oder vertikale Linien sind einfacher umzusetzen als diagonale Linien. Nutze eine Verzweigung mit mehreren elif-Abfragen.

Hinweis: Eine Auswahl an Farben findest du hier:

<https://trinket.io/docs/colors>



- d) Experimentiere weiter mit verknüpften Bedingungen. Welche Formen und Muster kannst du erstellen?

Wiederholungen

A3

- a) Entscheide welche der folgenden Codeblöcke genutzt werden können, um das Türschild rechts zu gestalten. Es gibt mehr als eine richtige Lösung.

- for counter in range (10):
 print('Kein Zutritt!!!')
- for counter in range (1, 11):
 print('Kein Zutritt!!!')
- for counter in range (1, 10):
 print('Kein Zutritt!!!')
- for counter in range (4, 9):
 print('Kein Zutritt!!!')
 print('Kein Zutritt!!!')
- for counter in range (11, 1):
 print('Kein Zutritt!!!')



- b) Folgender Code soll die nebenstehende Ausgabe erzeugen. Ergänze die fehlenden Werte (□) im Code und gib jeweils die Werte von hip_zaeahler und hurra_zaeahler in der Ausgabe an.

```
for hurra_zaeahler in range(□, □):
```

```
    for hip_zaeahler in range(□, □):
```

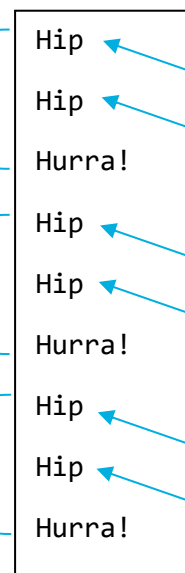
```
        print('Hip')
```

```
    print('Hurra!')
```

```
hurra_zaeahler = □
```

```
hurra_zaeahler = □
```

```
hurra_zaeahler = □
```



- c) Schreibe ein Programm, welches mit Hilfe von Wiederholungsanweisungen den folgenden Songtext generiert (ohne Satzzeichen). Orientiere dich an Aufgabenteil b).

„Lollipop lollipop. Oh lolli lolli lolli, lollipop, lollipop.
Oh lolli lolli lolli, lollipop, lollipop. Oh lolli lolli lolli, lollipop.“

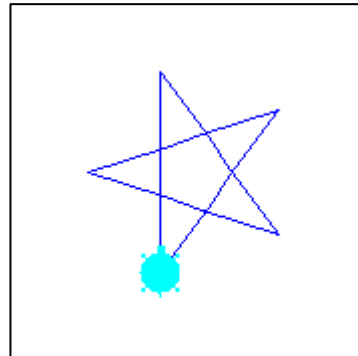
A4

Das folgende Programm zeichnet einen fünfzackigen Stern.

```
from gturtle import*
makeTurtle()

Groesse = 100
Spitzen = 5
Winkel = 180 - (180/Spitzen)

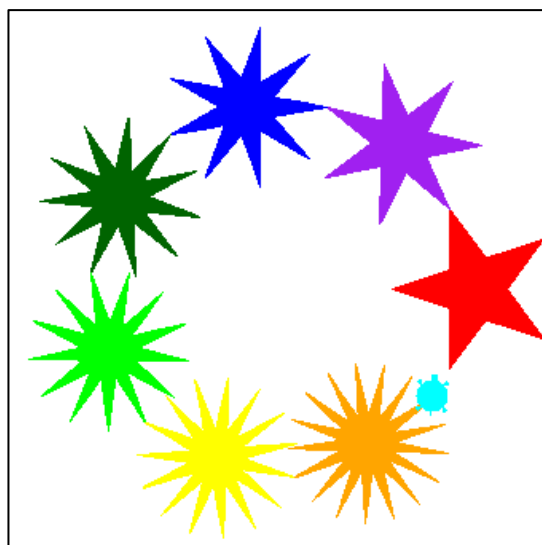
for i in range(Spitzen):
    forward(Groesse)
    left(Winkel)
```



- Ändere das Programm so ab, dass es Sterne mit mehr Spitzen zeichnet. Welche Einschränkung gibt es bei der Anzahl des Spitzen? Ändere die Größe der Sterne.
- Ergänze vor der for-Schleife den Befehl `startPath()` und hinter der for-Schleife den Befehl `fillPath()`. Teste die Änderungen.
- Lagere die Funktion in ein Unterprogramm Stern aus, welches die Parameter Groesse und Spitzen übergeben bekommt.
- Das Unterprogramm Stern wird mit folgendem Code aufgerufen. Welche Ausgabe generiert die Turtle?

```
for i in range (2, 5):
    Stern(100, 2*i+1)
    forward(100)
    left(50)
```

- Mit `setPenColor(...)` und `setFillColor(...)` kannst du die Farbe der Linien und der Füllung anpassen. Erweitere dein Programm so, dass der Nutzer für jeden Stern eine Farbe angeben kann.



Vertiefung

A5

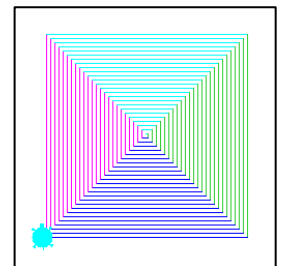
In dieser Aufgabe programmierst du ein einfaches Spiel. Um den Spielstand „gewonnen“ oder „verloren“ zu speichern, nutzt du eine sogenannte **Flag**. Dies ist eine Markierung, die entweder True oder False ist. An einer geeigneten Stelle kann die Flag überprüft werden und eine entsprechende Ausgabe generiert werden.

Das Spiel heißt „Zahlen raten“. Das Programm gibt intern eine Zahl vor, welche der Nutzer erraten muss. Dabei hat er nur eine vorgegebene Anzahl an Versuchen.

- Starte deine Programmierung damit, eine Flag für den Spielstand anzulegen und entsprechende Ausgaben zu generieren.
- Sorge als nächstes dafür, dass der Nutzer nach einer vorgegebenen Anzahl Versuche das Spiel automatisch verloren hat.
- Gib dem Nutzer nun die Möglichkeit in jedem Versuch eine neue Zahl einzugeben. Überprüfe die Eingabe und entscheide, ob der Nutzer gewonnen hat oder weiterspielen soll.
- Mit Hinweisen kannst du das Spiel für den Nutzer spannender gestalten.
- Verfeinere dein Programm nach deinen Vorstellungen. Lass zum Beispiel einen anderen Nutzer die zu erratene Zahl und die Anzahl der Versuche eingeben.

A6

In dieser Aufgabe schreibst du ein Programm, das nebenstehendes Muster erzeugt. Die Teilaufgaben helfen dir jeweils einen Teil des Programms umzusetzen.



- Bei der Funktion `range(...)` kannst du als dritten Parameter die Schrittweite angeben.
Betrachte folgenden Code. Welches Muster generiert die Turtle und welche Zahlen werden auf der Konsole ausgegeben?

```
from gturtle import*
makeTurtle()

for i in range(0, 200, 2):
    print(i)
    forward(i)
    right(90)
```

- Ändere den Code so ab, dass die horizontalen Linien eine andere Farbe haben als die vertikalen Linien.
Tipp: Entscheide die Farbe in Abhängigkeit von `i`. Mit der Bedingung `i % 4 == 0` kannst du überprüfen, ob `i` durch 4 teilbar ist.
- Nun mit vier Farben! Welche Gemeinsamkeiten haben die Zahlen, deren Linien jeweils dieselbe Farbe haben? Verknüpfe mehrere Bedingungen.